E0-253: Operating Systems Programming Assignment

Background

Physical memory is often over-committed by operating systems and hypervisors wherein the total memory demand of applications exceeds the memory capacity available in the system. This assignment is related to two of the techniques that are used to facilitate memory over-commitment. The first technique (referred to as swapping) uses pre-configured disk space to swap out (infrequently accessed) pages from memory. Demand paging loads these pages back into memory when they are accessed by the applications. The second technique (referred to as memory ballooning) is commonly used in virtualized environments. Under ballooning, the guest OS running inside a virtual machine is responsible for releasing some of its memory upon hypervisor request. It is important to note that under ballooning, the virtual machine itself is responsible for saving its data before releasing pages back to the hypervisor.

Assignment summary

In this assignment, you will implement application-level ballooning. However, unlike traditional ballooning, you will seek support from the OS to save your data before releasing memory. At a high-level, we require you to build two components — one in kernel space and one in user space. The user space component will implement a page replacement policy of your choice while the kernel space component will implement the necessary mechanisms.

In the kernel space component, you are responsible for checking the state of free memory in the system. When the amount of free memory falls below the specified threshold, you will send a signal SIGBALLOON to the application. Note that no such signal exists in the kernel today – you will create one for this assignment. In response to this signal, the application will issue one or more system calls asking for some of its pages to be swapped. The system call handler in the kernel will implement the actual page replacement mechanism.

In the user space component, you are free to implement any page replacement policy. Upon receiving the SIGBALLOON signal, your policy will first decide which page(s) it wants to swap out to disk. To choose an efficient page replacement policy, you can use the "idle page tracking" and "pagemap" infrastructure of Linux. It will help you in estimating the access patterns of your applications [3]. While we will not expose you to the source (except for some simple examples), you can assume that the application adheres to the basic principles of spatial and temporal locality.

Note: Note that only your main application will participate in ballooning via swapping. You need to disable swapping for all other applications in the system.

Getting started

You will build the kernel driver in Linux version 5.11.5. Download it as follows:

```
$ wget https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.11.5.tar.xz
```

The application you will modify is available on github. Clone it on your system as follows:

```
$ git clone https://github.com/csl-iisc/e0253-os-2021.git
```

Linux supports multiple page sizes. In this assignment, you will work only with base 4KB pages – disable Transparent Huge Pages manually or while compiling your kernel image. Your user space code must be implemented only in "main.c". Do not update or add any other files. Follow the README.md in github repository for more instructions.

Assignment-1 (Due date: 10/04/2021)

Kernel space: Implement a system call that an application can use to register itself with the ballooning subsystem in the kernel. Your system call is expected to do the following: (1) disable default swapping algorithm of

the kernel – no pages from other applications can be swapped after the system call returns. (2) set up a mechanism that will deliver the SIGBALLOON signal to the registered application. The signal must be delivered when the amount of free memory falls below 1GB.

User space: In the user space component: (1) register the application with the kernel's ballooning component using the system call as discussed above, and (2) set up a handler to receive the SIGBALLOON signal from the kernel. File "main.c" contains a variable named `nr_signals` – you are expected to increment it each time the signal is received in the application.

Deliverables: A single kernel patch that contains all your changes and "main.c" user space file.

Assignment-2 (Due date: 10/05/2021)

User space: In the user space component, implement a page replacement policy to decide page(s) to be swapped. For this, you will need to introduce one or more system calls to notify the kernel of page(s) that you want to swap. Keep in mind that swapped out pages can be accessed by the application at any point. You also need to ensure that demand loading these pages back in memory works as expected. It may require you to send more signals to the application and swap out other pages if free memory is less than 1GB. Optimize your implementation to minimize I/O activities. Follow the basic principles of spatial and temporal locality while optimizing your policy. After receiving the signal, the application has to respond with pages to swap within 10 seconds.

Kernel space: Implement the systems call(s) that will swap the application specified pages. There is no restriction on the design of system call interface – you can pass as many number or type of arguments as you wish.

Deliverables: A single kernel patch that contains all your changes and "main.c" user space file. The kernel patch must also include your changes from assignment-1.

Assignment-3 (Due date: 10/06/2021)

Kernel space: Extend your kernel implementation from assignment-2 to support 2MB transparent huge pages (THP). A user can enable THP by setting the flag "/sys/kernel/mm/transparent_hugepage/enabled" to "always". Depending on system configuration, state of memory fragmentation, and size or alignment of the virtual address regions, it is possible that some parts of the user memory are mapped with 2MB pages while the others are mapped with 4KB pages. Your implementation should handle all possible mapping scenarios. **Options:** While swapping out a specific address range, you may choose to swap out its 2MB pages as is (coarse-grained) or may split 2MB pages into 4KB pages to achieve fine-grained swapping at 4KB granularity.

User space: Extend the user-level page replacement policy to work with THP. You can modify the system call interface in any way you want (e.g., your policy may prefer coarse-grained or fine-grained swapping depending on access patterns – the system call can be used to pass these policy decisions to the kernel).

Deliverables: A single kernel patch that contains all your changes and "main.c" user space file. The kernel patch must also include your changes from assignment-1 and assignment-2.

Submission instructions

For each deliverable, you will submit an encrypted file. First, place all your files in a folder named as `#####_e0253_*`. The zip file must be named as `#####_e0253_*.zip`. Replace `#####` with the last five digits of your SR number and * with the assignment number. For example, for SR number 15943 and assignment-1, the zipped file must be named as `15943_e0253_1.zip`. The key for encrypting your zip file is provided at the end of this document – copy it to a file named `e0253.cert`. Once you have placed all files in the folder as specified above, you can produce the encrypted file to submit as follows:

```
$ zip -r 15943_e0253_1.zip 15943_e0253_1/
$ openssl smime -encrypt -binary -aes-256-cbc -in 15943_e0253_1.zip -out 15943_e0253_1.enc -outform
  DER e0253.cert
```

Upload encrypted files to the respective assignment directory in the following link:
`https://indianinstituteofscience-my.sharepoint.com/:f:/g/personal/vg_iisc_ac_in/EqFyDGKhx8dAn_ZM8i7lQyMB7IFcl4SC_ceslLMSswwJSA?e=GqIjWA`

How to generate a patch using git:

First, stage all modified files for a commit using "git add". Second, prepare a commit signed with your email id "git

commit -s -m". Third, fetch the commit id using "git log" (this will be the top-most commit id, say XXX). Finally, prepare a patch from your commit using "git format-patch".

```
$ git add include/linux/abc.h
$ git add mm/xyz.c
$ git commit -s -m "os assignment-1"
$ git log
$ git format-patch XXX -1
```

Evaluation:
Evaluation will be based on two key aspects:

1. **Functional correctness:** Implementation should work for arbitrary programs.
2. **Performance:** You should try to minimize the time spent in performing swap related IO, the number of times SIGBALLOON is delivered to the application and systems call invocations for swapping pages. The application must spend most of its time in executing the test cases.

References
[1] Understanding the Linux Kernel (3rd edition) by Daniel P. Bovet and Marco Cesati (Chapter 3 and 9).
[2] Understanding the Linux Virtual Memory Manager by Mel Gorman (Chapter 4, 6, 10, 11 and Appendix J, K in code commentary).
[3] Idle Page Tracking. `https://www.kernel.org/doc/html/latest/admin-guide/mm/idle_page_tracking.html`
[4] Pagemap, from the userspace perspective. `https://www.kernel.org/doc/Documentation/vm/pagemap.txt`
[5] Look at the following files in Linux source to get started:
`include/linux/mm_types.h`
`include/linux/mm.h`
`mm/swap.c`

-----BEGIN CERTIFICATE-----
MIIFazCCA1OgAwIBAgIUKcKY7jhF3V7fpuMLhtFZCGiXgbMwDQYJKoZIhvcNAQEL
BQAwRTELMAkGA1UEBhMCQVUxEzARBgNVBAgMClNvbWUtU3RhdGUxITAfBgNVBAoM
GEludGVybmV0IFdpZGdpdHMgUHR5IEx0ZDAeFw0yMTA0MDcwNDM2MzZaFw0yMjA0
MDcwNDM2MzZaMEUxCzAJBgNVBAYTAkFVMRMwEQYDVQQIDApTb211lLVN0YXRlMSEw
HwYDVQQKDBhJbnRlcm5ldCBXaWRnaXRzIFB0eSBMdGQwggIiMA0GCSqGSIb3DQEB
AQUAA4ICDwAwggIKAoICAQC6UCgD3aAyaoyUDnO0SItatV65K8zvfY5OvsddCmWk
0BM6enFDXifmOBF+b+glUfY/UASkd/a+RJw7qQQAqsVA/51LM1oUm341kbKcWpX8
umvvm2ohzXlAAhnq7tYE02jYeKCg6X0OQDKqe+FhzOaaX8/sHHzrlwpCCBtDQZAT
SG/HwSaUZrTq2R8Dqx3+qVlx4xdzh4r4gu2OKHzjBE6QJpBgkVjvJcQlY1ZDb0B6
L5I6fZbPZGW9QcJudoxzPEoF9s+aMkjsEBY1NPt92jAFwM0PtEojmWBp9vjT5ihx
9UE/f5EJYFg+IdppzfOguoFSE3FcVUTVRQnc1wlfu4OhOns65kCZ8pURpd9ZJ25c
6wPuqxbD/QdcAqT4NqrYtsA6a3T0bkNFHeh549T4Fv+cL3z1RN680no4dnnuJxv1
Mi5FRLbgPpdhG3vie6CarQZUDHQUmbcSLfM1ekVdis5aoo+JdsqWK0ViiEYHJQNu
st3YsKFS0KupDOUrGlVe/Qyl32A3ekYU9ljoM/5bsUbz/WgO0onRgxxLJLIT4sSn
Idzk40qfeBIoXUvyJ56T7+k20UkP/cfbI0yJZ4A5HSSosCHr7+K7dK2JmZra9Mxo
bNNa+1wtwItXgPU+I3L3WRJp+jssce5pxBqxUjs0S3dIKPdeSYwwl7DJMaOtX76G
7wIDAQABo1MwUTAdBgNVHQ4EFgQU/nQRK4TJxgLbW+xNlYc7MpYGbBYwHwYDVR0j
BBgwFoAU/nQRK4TJxgLbW+xNlYc7MpYGbBYwDwYDVR0TAQH/BAUwAwEB/zANBgkq
hkiG9w0BAQsFAAOCAgEAQkx8/nW7WLjJVTHp7zET8bsqe/gUxXOpnDH0JG8XN7j3
kf8dsPv6ZdLoCSqsnD9EZY6dTWwIB+gl2gABHUjz3SOFWQdm6vpeE7eBcepN/pg7
hxHgQqsb5hjXqjGgtWgy8PZ76jUAPbFOxMm6Xfa6dtYG/795Zt5qh5qdKhGAAHdo
K+J+ij+rnuzTI5JkZuBLnDIlZScbG0SQOpFsiGkZX/Tut8xLRvAaIC6VQUdG21GQ

```
sDZQHaWhOWXNtqzsnPQT8EIhCvyH9k2mrJjL6039439clC5OJH2HiAV8FZpxizWR
6gdoMcghFK07W/7kllOt38qpXkHkqGzSgyw6OcZjYfPkXMl7BXIlNduqzOOFhNjv
bdCfaNv5M7TKBN71KIBBls+HMHhjYiGYhh+XHxT+GyzbSqh1yU3td5r2LEsJROh5
gQnTi7T1KGgJCDoz2JVGLbD19NYgLnwfGz+u0Gdt6HYrBQlsJD0eb6tbxyEwC5pg
TIycXj1TY3mPCScjHUu2uYF38emgzcVWmrjmtHok8LUwSgrNSN0BMcoq3FjyWt0f
ar65iYlPFKkKNTYaMIoBpQki23ajSe+g4BXdvVh+kdSi32YZRLC5xRoTPJvmz7Vl
RR+3P2iaYCywri1tgSS0KMsJbiA69yUcoEaHD1XwB+U2hfn8JS0mIw0TiIfxHlw=
-----END CERTIFICATE-----
```